Melec

STEPPING & SERVO MOTOR CONTROLLER'S OPTION

MPL-16V5.0-PCIWXP
取扱説明書

C-870v1
C-871
C-872
C-873
C-874
C-874v1
C-875

本製品を使用する前に、この取扱説明書を良く読んで
十分に理解してください。
この取扱説明書は、いつでも取り出して読めるように
保管してください。

MN0056

目次

1. 概要

MPL-16v5.0-PCIWXPは、DOS/VパソコンのWindows上でPCI BUSを介して、弊社製ステッピング＆サーボモータコントローラボードを動作させるためのDLLベースのドライバ関数です。各関数は、次に示すBOARD CONTROLLER上のPORTのアクセス(読み出し/書き込み)及び、割り込み発生による処理を行う為のものです。BOARD CONTROLLER上のPORT、割り込み要求信号(RDYINT、CNTINT、DFLINT、IOINT)については、各BOARD CONTROLLERの取扱説明書を御覧ください。

(1)MCC05 PORT
  DRIVE COMMAND PORT
  DRIVE DATA1 PORT
  DRIVE DATA2 PORT
  DRIVE DATA3 PORT
  STATUS1 PORT
  STATUS2 PORT
  STATUS3 PORT
  STATUS4 PORT
  STATUS5 PORT
  COUNTER COMMAND PORT
  COUNTER DATA1 PORT
  COUNTER DATA2 PORT
  COUNTER DATA3 PORT
(2)汎用I/O PORT
(3)増設I/O PORT
(4)C-874v1 Z/A SENSOR割り付けPORT

2. ボードNo.の設定

当デバイスドライバを使用すると、最大10枚のBOARD CONTROLLERを制御することが可能です。複数のBOARD CONTROLLERをご使用の場合、各BOARD CONTROLLER上のボードNo.(S1 ロータリースイッチ)は、必ず異なる設定にしてください。

3. デバイスのオープンとクローズ

ユーザアプリケーションは、MCC05 PORTにアクセス前にボードNo.と軸を指定してデバイスをオープン(デバイスオープン関数)し、デバイスハンドルを受け取ります。以後、各関数を実行する際にこのデバイスハンドルを引数として渡します。このデバイスハンドルは、デバイスをクローズするまで有効です。ユーザアプリケーション終了時は、必ずデバイスをクローズしてください。クローズが行われていないと、以後正常に動作しません。

4. I/O PORT のオープンとクローズ

ユーザアプリケーションは、汎用I/O PORT、拡張I/O PORTにアクセス前にボードNo.とPORT No.を指定してI/O PORTをオープン(I/O PORTオープン関数、I/O PORTオープンEx関数)し、I/O PORTハンドルを受け取ります。以後、各関数を実行する際にこのI/O PORTハンドルを引数として渡します。このI/O PORTハンドルは、I/O PORTをクローズするまで有効です。ユーザアプリケーション終了時は、必ずI/O PORTをクローズしてください。クローズが行われていないと、以後正常に動作しません。

5. 割り込み

割り込みの通知方法は、次の2つが使用できます。

通知方法	説明
コールバック	割り込みが発生したときに、設定された関数をコールバックします。
メッセージ	割り込みが発生したときに、設定されたウィンドウのウィンドウプロシージャにメッセージをポストします。

5-1.割り込み処理の手順

ユーザアプリケーションで割り込みを使用する場合、次の手順で行ってください。
MPLを使用する前に短絡ソケットを使って各BOARD CONTROLLER上の使用するMCC05v2割り込み要求信号(RDYINT、CNTINT、DFLINT)と増設I/O割り込み要求信号(IOINT)の選択を行います。増設I/O PORTは、C-875の場合のみとなります。
詳細については、各BOARD CONTROLLER上の取扱説明書を御覧ください。

次の手順で割り込みに関する設定を行って下さい。
①割り込みINITIALIZE関数を実行してください。
複数のBOARD CONTROLLERを御使用の場合は、各BOARD CONTROLLER毎に割り込みINITIALIZE関数を実行してください。
割り込みINITIALIZE関数では、MCC05v2の次のCOMMANDと増設I/O INT SET PORTのINT ENABLE BITを使用し、
指定されたBOARD CONTROLLER上の全割り込み要求信号(RDYINT、CNTINT、DFLINT、IOINT)が出力されない設定にし、
その他の設定をデフォルトの値に設定します。

・SPEC INITIALIZE1 COMMAND
  DRIVE TYPE : L-TYPE
  LIMIT STOP TYPE : 即時停止
  MOTOR TYPE : STEPPING
  RDYINT TYPE : いかなる場合も出力しない

・PULSE COUNTER INITIALIZE COMMAND
  COMP STOP TYPE : 即時停止
  CNTINT OUTPUT TYPE : 各COMPARATORの検出状態をラッチ出力
  CNTINT LATCH TRIGGER TYPE : エッジラッチ
  COUNT CLOCK TYPE : MCC05v2のDRIVE PULSE(CWP,CCWP)で動作する
  COUNT PATTERN TYPE : EA,EB入力を1逓倍してカウントする
  AUTO CLEAR ENABLE : オートクリアを行わない
  RELOAD ENABLE : リロードを行わない
  COMP1 INT ENABLE : CNTINTを出力しない
  COMP1 STOP TYPE : 停止させない
  COMP2 INT ENABLE : CNTINTを出力しない
  COMP2 STOP TYPE : 停止させない
  COMP3 INT ENABLE : CNTINTを出力しない
  COMP3 STOP TYPE : 停止させない
  COMP4 INT ENABLE : CNTINTを出力しない
  COMP4 STOP TYPE : 停止させない
  COMP5 INT ENABLE : CNTINTを出力しない
  COMP5 STOP TYPE : 停止させない

・DFL COUNTER INITIALIZE COMMAND
  DFLCOMP STOP TYPE : 即時停止
  DFLINT OUTPUT TYPE : 各COMPARATORの検出状態をラッチ出力
  DFLINT LATCH TRIGGER TYPE : エッジラッチ
  COUNT CLOCK TYPE : MCC05v2のDRIVE PULSE(CWP,CCWP)とEA,EBからの外部クロックの偏差で動作する
  COUNT PATTERN TYPE : EA,EB入力を1逓倍してカウントする
  COMP1 INT ENABLE : DFLINTを出力しない
  COMP1 STOP TYPE : 停止させない
  COMP2 INT ENABLE : DFLINTを出力しない
  COMP2 STOP TYPE : 停止させない

・INT MASK COMMAND
  PLS COMP1 INT MASK : マスクする
  PLS COMP2 INT MASK : マスクする
  PLS COMP3 INT MASK : マスクする
  PLS COMP4 INT MASK : マスクする
  PLS COMP5 INT MASK : マスクする

DFL COMP1 INT MASK　　　　：マスクする
DFL COMP2 INT MASK　　　　：マスクする

割り込みINITIALIZE関数が実行されていないBOARD CONTROLLERでは、割り込みオープン関数は実行できません。

②MCC05v2割り込み要求信号(RDYINT、CNTINT、DFLINT)を使用する場合は、デバイスオープン関数でデバイスをオープンしてください。
増設I/O PORT割り込み要求信号(IOINT)を使用する場合は、I/O PORTオープンEx関数でI/O PORTをオープンしてください。

③BOARD CONTROLLER上の割り込み要求信号(RDYINT、CNTINT、DFLINT)の設定を行って下さい。
割り込み要求信号を使用する場合は、次のCOMMANDの設定DATAを、下記のように設定してください。

・RDYINTを使用する場合
SPEC INITIALIZE1 COMMAND
RDYINT TYPE：RDYINTの発生パターンを指定して下さい。

・CNTINTを使用する場合
PULSE COUNTER INITIALIZE COMMAND
使用するCOMPARE REGISTERの検出出力CNTINTを出力するに設定して下さい。
(COMP1 INT ENABLE、COMP2 INT ENABLE、COMP3 INT ENABLE、COMP4 INT ENABLE、COMP5 INT ENABLE)
CNTINT OUTPUT TYPE　　　　：各COMPARATORの検出状態をラッチ出力
CNTINT LATCH TRRIGER TYPE：エッジラッチ

・DFLINTを使用する場合
DFL COUNTER INITIALIZE COMMAND
使用する偏差COUNTER COMPARE REGISTERの検出出力DFLINTを出力するに設定して下さい。
(COMP1 INT ENABLE、COMP2 INT ENABLE)
DFLINT OUTPUT TYPE　　　　：各COMPARATORの検出状態をラッチ出力
DFLINT LATCH TRRIGER TYPE：エッジラッチ

・IOINTを使用する場合
特に行うことはありません。

割り込み要求信号を使用しない場合は、割り込み要求信号出力の設定を変更しないで下さい。
BOARD CONTROLLER上の割り込み要求信号を複数軸使用する場合は、割り込みオープン関数を実行する前に、全て設定して下さい。

④デバイス割り込みオープン関数(RDYINT、CNTINT、DFLINT)、I/O PORT割り込みオープン関数(IOINT)で割り込みをオープンします。デバイス割り込みオープン関数(I/O PORT割り込みオープン関数)にデバイスハンドル(I/O PORTハンドル)を引数で渡してください。
デバイス割り込みオープン関数では、指定されたデバイスがCNTINT、DFLINTを出力する設定なっている場合、MCC05v2の次のCOMMANDを使用し、そのデバイスのCNTINT、DFLINTのマスクを解除します。

・INT MASK COMMAND
PLS COMP1 INT MASK：マスクしない　　PLS COMP5 INT MASK：マスクしない
PLS COMP2 INT MASK：マスクしない　　DFL COMP1 INT MASK：マスクしない
PLS COMP3 INT MASK：マスクしない　　DFL COMP2 INT MASK：マスクしない
PLS COMP4 INT MASK：マスクしない

I/O PORT割り込みオープン関数では、増設I/O INT SET PORTのINT ENABLE、INT EDGE TYPEを設定しています。

(注1)割り込みINITIALIZE関数が実行されたボード上のデバイスでは、下記のCOMMANDで設定DATAを次のように指定した場合、エラーとなり、そのデバイスの全割り込み要求信号(RDYINT、CNTINT、DFLINT)が出力されないように設定されます。

・PULSE COUNTER INITIALIZE COMMAND
『CNTINTを出力する』に設定されている場合
CNTINT OUTPUT TYPE　　　　：各COMPARATORの検出状態をそのままスルーして出力
CNTINT LATCH TRRIGER TYPE：レベルラッチ

・DFL COUNTER INITIALIZE COMMAND
『DFLINTを出力する』に設定されている場合
DFLINT OUTPUT TYPE　　　　：各COMPARATORの検出状態をそのままスルーして出力
DFLINT LATCH TRRIGER TYPE：レベルラッチ

(注2)割り込みINITIALIZE関数が実行されたボード上のデバイスでは、INT MASK COMMANDは受け付けなくなります。

(注3)割り込みINITIALIZE関数は、アプリケーションの先頭で一度だけ実行してください。
割り込みINITIALIZE関数は、指定されたBOARD CONTROLLER上の全割り込み要求信号(RDYINT、CNTINT、DFLINT、IOINT)を出力されないに設定にし、他をデフォルトに設定しますが、同じアプリケーションでは、2度目以後、マルチプロセスでは他のプロセスで行われていた場合は、この処理を行いません。
このため、同じアプリケーションにて使用する割り込み信号が途中で変わる場合には、次の順序で処理を行ってください。

◎RDYINT、CNTINT、DFLINTの場合
a.割り込みINITIALIZE関数の実行
関数内で自動的に全割り込み要求信号が出力されない設定にされます。

b.デバイスオープン関数の実行
c.アプリケーションで使用する割り込み要求信号を設定します。
(SPEC1/PULSE COUNTER/DFL COUNTER INITIALIZE COMMAND使用)
d.デバイス割り込みオープン関数の実行
・各割り込みのコールバック、メッセージを設定します。
e.実行処理
f.割り込み要求信号の再設定
・使用しない割り込み要求信号を出力しないに設定し、使用する割り込み要求信号を出力するに設定します。
(SPEC1/PULSE COUNTER/DFL COUNTER INITIALIZE COMMAND使用)
g.実行処理

◎IOINTの場合

a.割り込みINITIALIZE関数の実行
・関数内で自動的にIOINTが出力されない設定にされます。
b.I/O PORTオープンEx関数の実行
c.I/O PORT割り込みオープン関数の実行
・INT ENABLE、EDGE TYPEを設定します。
・各割り込みのコールバック、メッセージを設定します。
d.実行処理
e.I/O PORT割り込みクローズ関数の実行
f.I/O PORT割り込みオープン関数の再実行
・INT ENABLE、EDGE TYPEを設定します。
・各割り込みのコールバック、メッセージを設定します。
g.実行処理

⑤ユーザアプリケーション終了時は、デバイス割り込みクローズ関数(I/O PORT割り込みクローズ関数)を使用して、割り込みをクローズしてください。クローズが行われていないと、以後正常に動作しません。

6. 関数説明

6-1.説明の見方

○○○○構造体 ← 構造体の名称

説　明

・・・・・・・・・・・・・・・・・・・・・・・・　→ 構造体の説明

書　式

C言語　・・　→ C言語で構造体を使用するときの定義

VB　・・　→ Visual Basicで、構造体を使用するときの定義

Delphi　・・　→ Delphiで、構造体を使用するときの定義

VB.NET　・・　→ Visual Basic.NETで、構造体を使用するときの定義

C#.NET　・・　→ C#.NETで、構造体を使用するときの定義

メンバ

・・・・・・・・・・・・・・・・・・・・・・・・　→ 構造体のメンバに格納される値の説明

○○○○関数 ← 関数の名称

機　能

・・・・・・・・・・・・・・・・・・・・・・・・　→ 関数の機能の説明

書　式

C言語　・・　→ C言語で、関数を使用するときの定義

VB　・・　→ Visual Basicで、関数を使用するときの定義

Delphi　・・　→ Delphiで、関数を使用するときの定義

VB.NET　・・　→ Visual Basic.NETで、関数を使用するときの定義

C#.NET　・・　→ C#.NETで、関数を使用するときの定義

引　数

・・・・・・・・・・・・・・・・・・・・・・・・　→ 関数の各引数に指定する値の説明

戻り値

・・・・・・・・・・・・・・・・・・・・・・・・　→ 関数の戻り値の説明

6-2. 構造体と関数

● 構造体一覧

構造体名	説明
RESULT構造体	関数を実行した結果を格納する構造体
データ構造体	デバイスに一括でアクセスするためのデータを格納する構造体
デバイス割り込み設定構造体	デバイス割り込みの各設定を行う為の構造体
I/O PORT割り込み設定構造体	I/O PORT割り込みの各設定を行う為の構造体
I/O PORT INT SET構造体	I/O PORT INT SETを行う為の構造体

● 関数一覧

関数名	機能
デバイスオープン関数	デバイスのオープン
デバイスクローズ関数	デバイスのクローズ
DRIVE COMMAND一括書き込み関数	DRIVE COMMAND, DATA1~3 PORTに一括書き込み
DRIVE DATA一括書き込み関数	DRIVE DATA1~3 PORTに一括書き込み
DRIVE COMMAND PORT書き込み関数	DRIVE COMMAND PORTに書き込み
DRIVE DATA1 PORT書き込み関数	DRIVE DATA1 PORTに書き込み
DRIVE DATA2 PORT書き込み関数	DRIVE DATA2 PORTに書き込み
DRIVE DATA3 PORT書き込み関数	DRIVE DATA3 PORTに書き込み
STATUS1 PORT読み出し関数	STATUS1 PORTの読み出し
STATUS2 PORT読み出し関数	STATUS2 PORTの読み出し
STATUS3 PORT読み出し関数	STATUS3 PORTの読み出し
STATUS4 PORT読み出し関数	STATUS4 PORTの読み出し
STATUS5 PORT読み出し関数	STATUS5 PORTの読み出し
DRIVE DATA一括読み出し関数	DRIVE DATA1~3 PORTの一括読み出し
DRIVE DATA1 PORT読み出し関数	DRIVE DATA1 PORTの読み出し
DRIVE DATA2 PORT読み出し関数	DRIVE DATA2 PORTの読み出し
DRIVE DATA3 PORT読み出し関数	DRIVE DATA3 PORTの読み出し
READY WAIT関数	パルスジェネレータがREADYになるまで待機
READY WAIT状態読み出し関数	READY WAIT関数の状態の読み出し
READY WAIT中止関数	READY WAIT関数のREADY待ちを中止
COUNTER COMMAND一括書き込み関数	COUNTER COMMAND, DATA1~3 PORTに一括書き込み
COUNTER COMMAND PORT書き込み関数	COUNTER COMMAND PORTに書き込み
COUNTER DATA1 PORT書き込み関数	COUNTER DATA1 PORTに書き込み
COUNTER DATA2 PORT書き込み関数	COUNTER DATA2 PORTに書き込み
COUNTER DATA3 PORT書き込み関数	COUNTER DATA3 PORTに書き込み
I/O PORTオープン関数	I/O PORTのオープン
I/O PORTオープン拡張関数	I/O PORTのオープン
I/O PORTクローズ関数	I/O PORTのクローズ
I/O PORT読み出し関数	I/O PORTの読み出し
I/O PORT書き込み関数	I/O PORTに書き込み
任意I/O PORT読み出し関数	任意I/O PORTの読み出し
任意I/O PORT読み出し拡張関数	任意I/O PORTの読み出し
任意I/O PORT書き込み関数	任意I/O PORTに書き込み
任意I/O PORT書き込み拡張関数	任意I/O PORTに書き込み
Z/A SENSOR割り付けPORT読み出し関数	Z/A SENSOR割り付けPORTの読み出し
Z/A SENSOR割り付けPORT書き込み関数	Z/A SENSOR割り付けPORTに書き込み
デバイス割り込みオープン関数	デバイス割り込みのオープン
I/O PORT割り込みオープン関数	I/O PORT割り込みのオープン
デバイス割り込みクローズ関数	デバイス割り込みのクローズ
I/O PORT割り込みクローズ関数	I/O PORT割り込みのクローズ
割り込みイニシャライズ関数	割り込みの初期化
データセット関数	32ビットデータのデータ構造体への格納
データゲット関数	データ構造体の32ビットデータの変換

RESULT構造体

説　明

関数を実行した結果が格納されます。

書　式

C言語
```
typedef struct MPL_TAG_S_RESULT {
    WORD  MPL_Result[4];
} MPL_S_RESULT;
```

VB
```
Type MPL_S_RESULT
    MPL_Result(1 To 4) As Interger
End Type
```

Delphi
```
MPL_S_RESULT = record
    MPL_Result: array[1..4] of WORD;
end;
```

VB.NET
```
Structure MPL_S_RESULT
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Public MPL_Result() As Short
    Public Sub Initialize()
        ReDim MPL_Result(4)
    End Sub
End Structure
```

C#.NET
```
[StructLayout(LayoutKind.Sequential)]
public struct MPL_S_RESULT{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst=4)] public ushort[] MPL_Result;
    public MPL_S_RESULT( ushort dummy ){
        MPL_Result = new ushort[4];
    }
}
```

メンバ

次に示すメンバは、C言語で表記しています。C言語の*MPL_Result[0]*～*MPL_Result[3]*は、Visual Basicでは*MPL_Result(1)*～*MPL_Result(4)*、Visual Basic.NETでは*MPL_Result(0)*～*MPL_Result(3)*、Delphiでは*MPL_Result[1]*～*MPL_Result[4]*、C#.NETでは*MPL_Result[0]*～*MPL_Result[3]*に対応します。

MPL_Result[0]　･･･ 実行された関数を示します。(値は10進表記です。)

値	関数名	値	関数名
10	デバイスオープン関数	51	DRIVE DATA1 PORT読み出し関数
12	デバイスクローズ関数	52	DRIVE DATA2 PORT読み出し関数
15	デバイス割り込みオープン関数	53	DRIVE DATA3 PORT読み出し関数
16	デバイス割り込みクローズ関数	70	I/O PORTオープン関数
17	割り込みINITIALIZE関数	71	I/O PORTオープン拡張関数
18	I/O PORT割り込みオープン関数	72	I/O PORTクローズ関数
19	I/O PORT割り込みクローズ関数	73	I/O PORT読み出し関数
20	DRIVE COMMAND PORT書き込み関数	74	I/O PORT書き込み関数
21	DRIVE DATA1 PORT書き込み関数	75	任意I/O PORT読み出し関数
22	DRIVE DATA2 PORT書き込み関数	76	任意I/O PORT書き込み関数
23	DRIVE DATA3 PORT書き込み関数	77	任意I/O PORT読み出し拡張関数
30	COUNTER COMMAND PORT書き込み関数	78	任意I/O PORT書き込み拡張関数
31	COUNTER DATA1 PORT書き込み関数	81	READY WAIT関数
32	COUNTER DATA2 PORT書き込み関数	82	READY WAIT状態読み出し関数
33	COUNTER DATA3 PORT書き込み関数	83	READY WAIT中止関数
41	STATUS1 PORT読み出し関数	90	DRIVE COMMAND一括書き込み関数
42	STATUS2 PORT読み出し関数	91	DRIVE DATA PORT一括読み出し関数
43	STATUS3 PORT読み出し関数	92	COUNTER COMMAND一括書き込み関数
44	STATUS4 PORT読み出し関数	93	DRIVE DATA一括書き込み関数
45	STATUS5 PORT読み出し関数	94	Z/A SENSOR割り付けPORT読み出し関数
		95	Z/A SENSOR割り付けPORT書き込み関数

MPL_Result[1]　･･･ 実行結果を示します。(値は10進表記です。)

値	実行結果
00:	関数の実行が正常に終了しました。
01:	カーネルドライバでエラーが発生しました。
02:	DLL内部でAPIエラーが発生しました。
03:	NULLポインタが指定されています。
04:	MplBdrv.sysがロードされていません。
05:	指定されたボード番号に誤りがあります。 又は、実行した関数は、指定されたボードでは、実行できません。
06:	軸又は、PORT番号の指定に誤りがあります。
07:	デバイスハンドルの内容が異常です。
08:	I/O PORTハンドルの内容が異常です。
09:	デバイスに空きがない為、オープン出来ません。
10:	I/O PORTに空きがない為、オープン出来ません。
11:	指定したデバイスは、オープンされていません。
12:	指定したI/O PORTは、オープンされていません。
13:	指定したデバイスは、すでにオープンされています。
14:	指定したI/O PORTは、すでにオープンされています。
15:	READY WAIT関数がTIME OVERで終了しています。
16:	WM_QUITメッセージを受信しました。
17:	READY WAIT中にREADY WAIT中止関数が実行されました。
18:	同一デバイスのREADY WAIT関数が複数同時に実行されました。
19:	ボードが１枚も検出できません。
20:	検出したボード枚数が10枚を超えました。
21:	指定されたボード番号に該当するボードがありません。
22:	ボード番号が重複しています。
23:	原因不明のエラーが発生しました。
24:	メッセージコードとユーザコールバック関数両方でNULL以外が指定されています。
25:	全てのメッセージコードとユーザコールバック関数が未使用になっています。
26:	指定したデバイス又は、I/O PORTの割り込みはオープンされていません。
27:	指定したデバイス又は、I/O PORTの割り込みはクローズされていません。
28:	指定したデバイスのRDYINTの設定が異なっています。
29:	指定したデバイスのCNTINTの設定が異なっています。
30:	割り込みが初期状態に設定されていません。
31:	割り込みINITIALIZE関数が実行された為、このコマンドは実行できません。
32:	I/O PORT INT SET構造体の内容が設定範囲を超えています。
33:	指定したI/O PORTは、割り込みを扱えません。

MPL_Result[2]　･･･ 将来の拡張用です。
MPL_Result[3]　･･･ 将来の拡張用です。

データ構造体

説　明

データを一括で読み書きするときに使用します。

● データを一括で読み書きするとき
・DRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTのデータを一括で書き込むとき
・DRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTのデータを一括で読み出すとき
・COUNTER DATA1 PORT、COUNTER DATA2 PORT、COUNTER DATA3 PORTのデータを一括で書き込むとき

書　式

C言語
```
typedef struct MPL_TAG_S_DATA {
    WORD MPL_Data[4];
} MPL_S_DATA;
```

VB
```
Type MPL_S_DATA
    MPL_Data(1 To 4) As Integer
End Type
```

Delphi
```
MPL_S_DATA = record
    MPL_Data: array[1..4] of WORD;
end;
```

VB.NET
```
Structure MPL_S_DATA
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Public MPL_Data() As Short
    Public Sub Initialize()
        ReDim MPL_Data(4)
    End Sub
End Structure
```

C#.NET
```
[StructLayout(LayoutKind.Sequential)]
public struct MPL_S_DATA{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst=4)] public ushort[] MPL_Data;
    public MPL_S_DATA( ushort dummy ){
        MPL_Data = new ushort[4];
    }
}
```

メンバ

次に示すメンバは、C言語で表記しています。C言語の*MPL_Data[0]*～*MPL_Data[3]*は、Visual Basicでは*MPL_Data(1)*～*MPL_Data(4)*、Visual Basic.NETでは*MPL_Data(0)*～*MPL_Data(3)*、Delphiでは*MPL_Data[1]*～*MPL_Data[4]*、C#.NETでは、*MPL_Data[0]*～*MPL_Data[3]*に対応します。

MPL_Data[0] ･･･ DRIVE DATA1 PORT、COUNTER DATA1 PORTのいずれかの内容を格納します。
MPL_Data[1] ･･･ DRIVE DATA2 PORT、COUNTER DATA2 PORTのいずれかの内容を格納します。
MPL_Data[2] ･･･ DRIVE DATA3 PORT、COUNTER DATA3 PORTのいずれかの内容を格納します。
MPL_Data[3] ･･･ 将来の拡張用です。

デバイス割り込み設定構造体

説　明

デバイス割り込みの各設定を行う為の構造体です。

書　式

C言語

```
typedef struct MPL_TAG_S_INT_CONFIG {
    UINT RdyInt_Message;
    UINT CntInt_Message;
    UINT Reserved_Message;
    PLPMPLCALLBACK RdyInt_CallBack;
    PLPMPLCALLBACK CntInt_CallBack;
    PLPMPLCALLBACK Reserved_CallBack;
    DWORD RdyInt_User;
    DWORD CntInt_User;
    DWORD Reserved_User;
} MPL_S_INT_CONFIG;
```

Delphi

```
MPL_S_INT_CONFIG = record
    RdyInt_Message:DWORD;
    CntInt_Message:DWORD;
    Reserved_Message:DWORD;
    RdyInt_CallBack:Pointer;
    CntInt_CallBack:Pointer;
    Reserved_CallBack:Pointer;
    RdyInt_User:DWORD;
    CntInt_User:DWORD;
    Reserved_User:DWORD;
end;
```

VB.NET

```
<StructLayout(LayoutKind.Sequential)> _
Public Structure MPL_S_INT_CONFIG
    Public RdyInt_Message As Integer
    Public CntInt_Message As Integer
    Public Reserved_Message As Integer
    Public RdyInt_CallBack As PLPMPLCALLBACK
    Public CntInt_CallBack As PLPMPLCALLBACK
    Public Reserved_CallBack As PLPMPLCALLBACK
    Public RdyInt_User As Integer
    Public CntInt_User As Integer
    Public Reserved_User As Integer
End Structure
```

C#.NET

```
[StructLayout(LayoutKind.Sequential)]
public struct MPL_S_INT_CONFIG{
    public uint            RdyInt_Message;
    public uint            CntInt_Message;
    public uint            Reserved_Message;
    public PLPMPLCALLBACK  RdyInt_CallBack;
    public PLPMPLCALLBACK  CntInt_CallBack;
    public PLPMPLCALLBACK  Reserved_CallBack;
    public uint            RdyInt_User;
    public uint            CntInt_User;
    public uint            Reserved_User;
}
```

メンバ

RdyInt_Message ··· RDYINT割り込み発生時に送出するメッセージコードを指定します。
メッセージポストを行わない場合には、WM_NULL(0)を指定してください。

CntInt_Message ··· CNTINT、DFLINT割り込み発生時に送出するメッセージコードを指定します。
メッセージポストを行わない場合には、WM_NULL(0)を指定してください。

Reserved_Message ··· 将来の拡張用です。WM_NULL(0)を指定してください。

RdyInt_CallBack ··· RDYINT割り込み発生時に呼び出されるユーザ・コールバック関数のアドレスを指定します。
コールバックを行わない場合には、0を指定してください。

CntInt_CallBack ··· CNTINT、DFLINT割り込み発生時に呼び出されるユーザ・コールバック関数のアドレスを
指定します。コールバックを行わない場合には、0を指定してください。

Reserved_CallBack ··· 将来の拡張用です。コールバックを行わない場合には、NULLを指定してください。

RdyInt_User ··· RDYINT割り込み発生時に呼び出されるユーザ・コールバック関数へ引き渡す
ユーザ・データを指定します。

CntInt_User ··· CNTINT、DFLINT割り込み発生時に呼び出されるユーザ・コールバック関数へ引き渡す
ユーザ・データを指定します。

Reserved_User ··· 将来の拡張用です。

I/O PORT割り込み設定構造体

説　明

I/O PORT割り込みの各設定を行う為の構造体です。

書　式

C言語
```
typedef struct MPL_TAG_S_IO_PORT_INT_CONFIG {
    UINT IoInt_Message;
    UINT Reserved_Message;
    PLPMPLCALLBACK IoInt_CallBack;
    PLPMPLCALLBACK Reserved_CallBack;
    DWORD IoInt_User;
    DWORD Reserved_User;
} MPL_S_IO_PORT_INT_CONFIG_INFO;
```

Delphi
```
MPL_S_IO_PORT_INT_CONFIG_INFO = record
    IoInt_Message:DWORD;
    Reserved_Message:DWORD;
    IoInt_CallBack:Pointer;
    Reserved_CallBack:Pointer;
    IoInt_User:DWORD;
    Reserved_User:DWORD;
end;
```

VB.NET
```
<StructLayout(LayoutKind.Sequential)> _
Public Structure MPL_S_IO_PORT_INT_CONFIG_INFO
    Public IoInt_Message As Integer
    Public Reserved_Message As Integer
    Public IoInt_CallBack As PLPMPLCALLBACK
    Public Reserved_CallBack As PLPMPLCALLBACK
    Public IoInt_User As Integer
    Public Reserved_User As Integer
End Structure
```

C#.NET
```
[StructLayout(LayoutKind.Sequential)]
public struct MPL_S_IO_PORT_INT_CONFIG_INFO{
    public uint             IoInt_Message;
    public uint             Reserved_Message;
    public PLPMPLCALLBACK   IoInt_CallBack;
    public PLMPLCALLBACK    Reserved_CallBack;
    public uint             IoInt_User;
    public uint             Reserved_User;
}
```

メンバ

IoInt_Message ･･･ IOINT割り込み発生時に送出するメッセージコードを指定します。
メッセージポストを行わない場合には、WM_NULLを指定してください。

Reserved_Message ･･･ 将来の拡張用です。WM_NULLを指定してください。

IoInt_CallBack ･･･ IOINT割り込み発生時に呼び出されるユーザ・コールバック関数のアドレスを指定します。
コールバックを行わない場合には、NULLを指定してください。

Reserved_CallBack ･･･ 将来の拡張用です。

IoInt_User ･･･ IOINT割り込み発生時に呼び出されるユーザ・コールバック関数へ引き渡す
ユーザ・データを指定します。

Reserved_User ･･･ 将来の拡張用です。

I/O PORT INT SET構造体

説　明

I/O PORT INT SETを行う為の構造体です。

書　式

C言語
```
typedef struct MPL_TAG_S_IO_PORT_INT_SET {
    WORD IntEnable1;
    WORD EdgeType1;
    WORD IntEnable2;
    WORD EdgeType2;
} MPL_S_IO_PORT_INT_SET;
```

Delphi
```
MPL_S_IO_PORT_INT_SET = record
    IntEnable1:WORD;
    EdgeType1WORD;
    IntEnable2:WORD;
    EdgeType2;WORD;
end;
```

VB.NET
```
<StructLayout(LayoutKind.Sequential)> _
Public Structure MPL_S_IO_PORT_INT_SET
    Public IntEnable1 As Short
    Public EdgeType1 As Short
    Public IntEnable2 As Short
    Public EdgeType2 As Short
End Structure
```

C#.NET
```
[StructLayout(LayoutKind.Sequential)]
public struct MPL_S_IO_PORT_INT_SET{
    public ushort   IntEnable1;
    public ushort   EdgeType1;
    public ushort   IntEnable2;
    public ushort   EdgeType2;
}
```

メンバ

IntEnable1 ･･･ I/O PORTの割り込み信号(IN10/IN30)のINT ENABLEを指定します。
0:割り込み発生禁止
1:割り込み発生許可

EdgeType1 ･･･ I/O PORTの割り込み信号(IN10/IN30)のEDGE TYPEを指定します。
0:立ち下がりエッジ(入力ON時)
1:立ち上がりエッジ(入力OFF時)

IntEnable2 ･･･ I/O PORTの割り込み信号(IN20/IN40)のINT ENABLEを指定します。
0:割り込み発生禁止
1:割り込み発生許可

EdgeType2 ･･･ I/O PORTの割り込み信号(IN20/IN40)のEDGE TYPEを指定します。
0:立ち下がりエッジ(入力ON時)
1:立ち上がりエッジ(入力OFF時)

ユーザコールバック関数

書　式

C言語　void CallBackProc(DWORD Status, DWORD UserData);

Delphi　procedure CallBackProc(Status:DWORD; UserData:DWORD);

VB.NET　Sub CallBackProc(ByVal Status As Integer, ByVal UserData As Integer)

C#.NET　void CallBackProc(uint Status, uint UserData);

メンバ

Status　･･･　・RDYINT信号によって割り込み要求が行われた場合は、STATUS1 PORTの内容を示します。
Statusの内容は下位8Bitが有効です。
・CNTINT信号又は、DFLINT信号によって割り込み要求が行われた場合は、
STATUS3 PORTの内容を示します。Statusの内容は下位8Bitが有効です。
・IOINT信号によって割り込み要求が行われた場合は、
IN10_20INT STATUS PORT、IN30_40INT STATUS PORTの内容を示します。
Statusの内容は8Bitが有効です。
STATUS1 PORT、STATUS3 PORT、IN10_20INT STATUS PORT、IN30_40INT STATUS PORTについては、
各BOARD CONTROLLERの取扱説明書を御参照下さい。

UserData ･･･ デバイス割り込み設定構造体又は、I/O PORT割り込み設定構造体で指定したユーザ・データです。

デバイスオープン関数

機　能

指定されたボード番号、軸で、デバイスをオープンし、引数*phDev*で示される変数にデバイスハンドルを格納します。

書　式

C言語
```
BOOL MPL_BOpen(HWND hWnd, WORD BoardNo, WORD Axis, DWORD FAR *phDev,
MPL_S_RESULT FAR *psResult);
```

VB
```
Function MPL_BOpen(ByVal hWnd As Long, ByVal BoardNo As Integer, ByVal Axis As Integer,
phDev As Long, psResult As MPL_S_RESULT) As Boolean
```

Delphi
```
function MPL_BOpen(hWnd:DWORD; BoardNo:WORD; Axis:WORD; var phDev: DWORD;
var psResult:MPL_S_RESULT):Boolean;
```

VB.NET
```
Function MPL_BOpen(ByVal hWnd As Integer, ByVal BoardNo As Short, ByVal Axis As Short,
ByRef phDev As Integer, ByRef psResult As MPL_S_RESULT) As Boolean
```

C#.NET
```
bool MPL.BOpen(uint hWnd, ushort BoardNo, ushort Axis, ref uint phDev,
ref MPL_S_RESULT psResult);
```

引　数

hWnd ・・・ 割り込み発生時に送出するメッセージのポスト先ウィンドウハンドルを指定します。Visual Basicをご使用の場合は、0を指定して下さい。
メッセージポストを行わない場合には、NULLを指定して下さい。

BoardNo ・・・ ボード番号を指定します。0～9のいずれかになります。

Axis ・・・ ボード上の軸を指定します。

C-870v1、C-871、C-874、C-874v1、C-875の場合

引数*Axis* の値	軸	引数*Axis* の値	軸
MPL_X	X軸	MPL_A	A軸
MPL_Y	Y軸	MPL_B	B軸
MPL_Z	Z軸	MPL_C	C軸

C-872、C-873の場合

引数*Axis* の値	軸	引数*Axis* の値	軸
MPL_X1	X1軸	MPL_X2	X2軸
MPL_Y1	Y1軸	MPL_Y2	Y2軸
MPL_Z1	Z1軸	MPL_Z2	Z2軸
MPL_A1	A1軸	MPL_A2	A2軸
MPL_B1	B1軸	MPL_B2	B2軸
MPL_C1	C1軸	MPL_C2	C2軸

phDev ・・・ デバイスハンドルが格納される変数のポインタを指定します。

psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

デバイスクローズ関数

機　能

指定されたデバイスをクローズします。

書　式

C言語　BOOL MPL_BClose(DWORD *hDev*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BClose(ByVal *hDev* As Long, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BClose(*hDev*: DWORD; var *psResult*: MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BClose(ByVal *hDev* As Integer, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET bool MPL.BClose(uint *hDev*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE COMMAND一括書き込み関数

機　能

指定されたデバイスのDRIVE COMMAND PORT、DRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTにコマンドコード、データを一括書き込みします。

書　式

C言語　BOOL MPL_IWDrive(DWORD *hDev*, WORD *Cmd*, MPL_S_DATA FAR **psData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_IWDrive(ByVal *hDev* As Long, ByVal *Cmd* As Integer, *psData* As MPL_S_DATA, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_IWDrive(*hDev*:DWORD; *Cmd*:WORD; var *psData*:MPL_S_DATA; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_IWDrive(ByVal *hDev* As Integer, ByVal *Cmd* As Short, ByRef *psData* As MPL_S_DATA, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.IWDrive(uint *hDev*, ushort Cmd, ref MPL_S_DATA *psData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
Cmd　･･･ 書き込むコマンドコードを指定します。
psData　･･･ 書き込むデータが格納されているデータ構造体のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE DATA一括書き込み関数

機　能

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTにデータを一括書き込みします。

書　式

C言語　BOOL MPL_IWData(DWORD *hDev*, MPL_S_DATA FAR **psData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_IWData(ByVal *hDev* As Long, *psData* As MPL_S_DATA, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_IWData(*hDev*:DWORD; var *psData*:MPL_S_DATA; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_IWData(ByVal *hDev* As Integer, ByRef *psData* As MPL_S_DATA, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.IWData(uint *hDev*, ref MPL_S_DATA *psData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　　・・・ デバイスハンドルを指定します。
psData　　・・・ 書き込むデータが格納されているデータ構造体のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE COMMAND PORT書き込み関数

機　能

指定されたデバイスのDRIVE COMMAND PORTにコマンドコードを書き込みます。

書　式

C言語　BOOL MPL_BWDriveCommand(DWORD *hDev*, WORD FAR **pCmd*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWDriveCommand(ByVal *hDev* As Long, *pCmd* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWDriveCommand(*hDev*:DWORD, var *pCmd*:WORD, var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BWDriveCommand(ByVal *hDev* As Integer, ByRef *pCmd* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWDriveCommand(uint *hDev*, ref ushort *pCmd*, ref MPL_S_RESULT *psResult*);

引　数

hDev　　・・・ デバイスハンドルを指定します。
pCmd　　・・・ 書き込むコマンドコードが格納されている変数のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE DATA1 PORT書き込み関数

機　能

指定されたデバイスのDRIVE DATA1 PORTにデータを書き込みます。

書　式

C言語　BOOL MPL_BWDriveData1(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWDriveData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWDriveData1(*hDev*:DWORD, var *pData*:WORD, var *psResult*:MPL_S_RESULT): Boolean;

VB.NET　Function MPL_BWDriveData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWDriveData1(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pData　･･･ 書き込むデータが格納されている変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE DATA2 PORT書き込み関数

機　能

指定されたデバイスのDRIVE DATA2 PORTにデータを書き込みます。

書　式

C言語　BOOL MPL_BWDriveData2(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWDriveData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWDriveData2(*hDev*:DWORD, var *pData*:WORD, var *psResult*:MPL_S_RESULT): Boolean;

VB.NET　Function MPL_BWDriveData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWDriveData2(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pData　･･･ 書き込むデータが格納されている変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE DATA3 PORT書き込み関数

機　能

指定されたデバイスのDRIVE DATA3 PORTにデータを書き込みます。

書　式

C言語　BOOL MPL_BWDriveData3(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWDriveData3(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWDriveData3(*hDev*: DWORD, var *pData*: WORD, var *psResult*: MPL_S_RESULT): Boolean;

VB.NET　Function MPL_BWDriveData3(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWDriveData3(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pData　･･･ 書き込むデータが格納されている変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

STATUS1 PORT読み出し関数

機　能

指定されたデバイスのSTATUS1 PORTを読み出します。

書　式

C言語　BOOL MPL_BRStatus1(DWORD *hDev*, WORD FAR **pStatus*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BRStatus1(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRStatus1(*hDev*: DWORD; var *pStatus*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BRStatus1(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRStatus1(uint *hDev*, ref ushort *pStatus*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pStatus　･･･ 読み出した内容が格納される変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

STATUS2 PORT読み出し関数

機　能

指定されたデバイスのSTATUS2 PORTを読み出します。

書　式

C言語　BOOL MPL_BRStatus2(DWORD *hDev*, WORD FAR **pStatus*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BRStatus2(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRStatus2(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BRStatus2(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRStatus2(uint *hDev*, ref ushort *pStatus*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pStatus　･･･ 読み出した内容が格納される変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

STATUS3 PORT読み出し関数

機　能

指定されたデバイスのSTATUS3 PORTを読み出します。

書　式

C言語　BOOL MPL_BRStatus3(DWORD *hDev*, WORD FAR **pStatus*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BRStatus3(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRStatus3(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BRStatus3(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRStatus3(uint *hDev*, ref ushort *pStatus*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pStatus　･･･ 読み出した内容が格納される変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

STATUS4 PORT読み出し関数

機　能

指定されたデバイスのSTATUS4 PORTを読み出します。

書　式

C言語　BOOL MPL_BRStatus4(DWORD *hDev*, WORD FAR **pStatus*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BRStatus4(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRStatus4(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BRStatus4(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRStatus4(uint *hDev*, ref ushort *pStatus*, ref MPL_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pStatus　・・・ 読み出した内容が格納される変数のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

STATUS5 PORT読み出し関数

機　能

指定されたデバイスのSTATUS5 PORTを読み出します。

書　式

C言語　BOOL MPL_BRStatus5(DWORD *hDev*, WORD FAR **pStatus*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BRStatus5(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRStatus5(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BRStatus5(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRStatus5(uint *hDev*, ref ushort *pStatus*, ref MPL_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pStatus　・・・ 読み出した内容が格納される変数のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE DATA一括読み出し関数

機　能

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTを一括読み出しします。

書　式

C言語　BOOL MPL_IRDrive(DWORD *hDev*, MPL_S_DATA FAR **psData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_IRDrive(ByVal *hDev* As Long, *psData* As MPL_S_DATA, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_IRDrive(*hDev*:DWORD; var *psData*:MPL_S_DATA; var *psResult*:MPL_S_RESULT): Boolean;

VB.NET　Function MPL_IRDrive(ByVal *hDev* As Integer, ByRef *psData* As MPL_S_DATA, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.IRDrive(uint *hDev*, ref MPL_S_DATA *psData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
psData　･･･ 読み出した内容が格納されるデータ構造体のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE DATA1 PORT読み出し関数

機　能

指定されたデバイスのDRIVE DATA1 PORTを読み出します。

書　式

C言語　BOOL MPL_BRDriveData1(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BRDriveData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRDriveData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BRDriveData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRDriveData1(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pData　･･･ 読み出した内容が格納される変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE DATA2 PORT読み出し関数

機　能

指定されたデバイスのDRIVE DATA2 PORTを読み出します。

書　式

C言語　BOOL MPL_BRDriveData2(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BRDriveData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRDriveData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BRDriveData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRDriveData2(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pData　・・・ 読み出した内容が格納される変数のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

DRIVE DATA3 PORT読み出し関数

機　能

指定されたデバイスのDRIVE DATA3 PORTを読み出します。

書　式

C言語　BOOL MPL_BRDriveData3(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BRDriveData3(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRDriveData3(*hDev*:DWORD; var *pData*:WORD; var *psResult*: MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BRDriveData3(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRDriveData3(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pData　・・・ 読み出した内容が格納される変数のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

READY WAIT関数

機　能

指定されたデバイス(MCC05v2)がREADY(STATUS1 PORTのBUSY=0)になるまで待機します。最大待ち時間を超えるとエラー終了します。

書　式

C言語　BOOL MPL_BWaitDriveCommand(DWORD *hDev*, WORD *WaitTime*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWaitDriveCommand(ByVal *hDev* As Long, ByVal *WaitTime* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWaitDriveCommand(*hDev*:DWORD; *WaitTime*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BWaitDriveCommand(ByVal *hDev* As Integer, ByVal *WaitTime* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWaitDriveCommand(uint *hDev*, ushort *WaitTime*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
WaitTime　･･･ 最大待ち時間を1ms単位で設定します。0を指定するとREADYになるまで無限に待機します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

READY WAIT状態読み出し関数

機　能

指定されたデバイスのREADY WAIT関数の状態を返します。

書　式

C言語　BOOL MPL_BIsWait(DWORD *hDev*, WORD FAR **pWaitSts*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BIsWait(ByVal *hDev* As Long, *pWaitSts* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BIsWait(*hDev*:DWORD; var *pWaitSts*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BIsWait(ByVal *hDev* As Integer, ByRef *pWaitSts* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BIsWait(uint *hDev*, ref ushort *pWaitSts*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pWaitSts　･･･ READY WAIT関数の状態が格納される変数のポインタを指定します。

格納される値	状態
0	READY WAIT関数を実行していません。
1	READY WAIT関数の実行中です。

psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

READY WAIT中止関数

機　能

指定されたデバイスのREADY WAIT関数のREADY待ちを中止します。

書　式

C言語　BOOL MPL_BBreakWait(DWORD *hDev*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BBreakWait(ByVal *hDev* As Long, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BBreakWait(*hDev*:DWORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BBreakWait(ByVal *hDev* As Integer, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BBreakWait(uint *hDev*, ref MPL_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

COUNTER COMMAND一括書き込み関数

機　能

指定されたデバイスのCOUNTER COMMAND PORT、COUNTER DATA1 PORT、COUNTER DATA2 PORT、COUNTER DATA3 PORTにコマンドコード、データを一括書き込みします。

書　式

C言語　BOOL MPL_IWCounter(DWORD *hDev*, WORD *Cmd*, MPL_S_DATA FAR **psData*,
MPL_S_RESULT FAR **psResult*);

VB　Function MPL_IWCounter(ByVal *hDev* As Long, ByVal *Cmd* As Integer, *psData* As MPL_S_DATA,
psResult As MPL_S_RESULT) As Boolean

Delphi　function MPL_IWCounter(*hDev*:DWORD; *Cmd*:WORD; var *psData*:MPL_S_DATA; var *psResult*:MPL_S_RESULT)
:Boolean;

VB.NET　Function MPL_IWCounter(ByVal *hDev* As Integer, ByVal *Cmd* As Short, ByRef *psData* As MPL_S_DATA,
ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.IWCounter(uint *hDev*, ushort Cmd, ref MPL_S_DATA psData, ref MPL_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
Cmd　・・・ 書き込むコマンドコードを指定します。
psData　・・・ 書き込むデータが格納されているデータ構造体のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

COUNTER COMMAND PORT書き込み関数

機　能

指定されたデバイスのCOUNTER COMMAND PORTにコマンドコードを書き込みます。

書　式

C言語　BOOL MPL_BWCounterCommand(DWORD *hDev*, WORD FAR **pCmd*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWCounterCommand(ByVal *hDev* As Long, *pCmd* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWCounterCommand(*hDev*:DWORD; var *pCmd*:WORD; var *psResult*:MPL_S_RESULT): Boolean;

VB.NET　Function MPL_BWCounterCommand(ByVal *hDev* As Integer, ByRef *pCmd* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWCounterCommand(uint *hDev*, ref ushort *pCmd*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pCmd　･･･ 書き込むコマンドコードが格納されている変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

COUNTER DATA1 PORT書き込み関数

機　能

指定されたデバイスのCOUNTER DATA1 PORTにデータを書き込みます。

書　式

C言語　BOOL MPL_BWCounterData1(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWCounterData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWCounterData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BWCounterData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWCounterData1(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pData　･･･ 書き込むデータが格納されている変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

COUNTER DATA2 PORT書き込み関数

機　能

指定されたデバイスのCOUNTER DATA2 PORTにデータを書き込みます。

書　式

C言語　BOOL MPL_BWCounterData2(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWCounterData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWCounterData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BWCounterData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWCounterData2(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pData　･･･ 書き込むデータが格納されている変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

COUNTER DATA3 PORT書き込み関数

機　能

指定されたデバイスのCOUNTER DATA3 PORTにデータを書き込みます。

書　式

C言語　BOOL MPL_BWCounterData3(DWORD *hDev*, WORD FAR **pData*, MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BWCounterData3(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWCounterData3(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BWCounterData3(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWCounterData3(uint *hDev*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hDev　･･･ デバイスハンドルを指定します。
pData　･･･ 書き込むデータが格納されている変数のポインタを指定します。
psResult　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

I/O PORTオープン関数

機　能

ボード番号を指定して、I/O PORTをオープンし、phPortで示される変数にI/O PORTハンドルを格納します。
C-872の場合、汎用I/O PORT1をオープンします。

書　式

C言語　BOOL MPL_BPortOpen(HWND *hWnd*, WORD *BoardNo*, DWORD FAR* *phPort*, MPL_S_RESULT FAR* *psResult*);

VB　Function MPL_BPortOpen(ByVal *hWnd* As Long, ByVal *BoardNo* As Integer, *phPort* As Long, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BPortOpen(*hWnd*:DWORD; *BoardNo*:WORD; var *phPort*:DWORD; var *psResult*:MPL_S_RESULT): Boolean;

VB.NET　Function MPL_BPortOpen(ByVal *hWnd* As Integer, ByVal *BoardNo* As Short, ByRef *phPort* As Integer, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BPortOpen(uint *hWnd*, ushort *BoardNo*, ref uint *phPort*, ref MPL_S_RESULT *psResult*);

引　数

hWnd ・・・ 割り込み発生時に送出するメッセージのポスト先ウィンドウハンドルを指定します。
Visual Basicをご使用の場合は、0を指定して下さい。
メッセージポストを行わない場合には、NULLを指定して下さい。
BoardNo ・・・ ボード番号を指定します。0～9のいずれかになります。
phPort ・・・ I/O PORTハンドルが格納される変数のポインタを指定します。
psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

I/O PORTオープン拡張関数

機　能

ボード番号、PORT番号を指定して、汎用I/O PORT又は、増設I/O PORTをオープンし、phPortで示される変数に
I/O PORTハンドルを格納します。

書　式

C言語　BOOL MPL_BPortOpenEx(HWND *hWnd*, WORD *BoardNo*, WORD *PortNo*, DWORD FAR* *phPort*,
MPL_S_RESULT FAR **psResult*);

VB　Function MPL_BPortOpenEx(ByVal *hWnd* As Long, ByVal *BoardNo* As Integer, ByVal *PortNo* As Integer,
phPort As Long, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BPortOpenEx(*hWnd*:DWORD; *BoardNo*:WORD; *PortNo*:WORD; var *phPort*:DWORD;
var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BPortOpenEx(ByVal *hWnd* As Integer, ByVal *BoardNo* As Short, ByVal *PortNo* As Short,
ByRef *phPort* As Integer, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BPortOpenEx(uint *hWnd*, ushort *BoardNo*, ushort *PortNo*, ref uint *phPort*,
ref MPL_S_RESULT *psResult*);

引　数

hWnd　・・・ 割り込み発生時に送出するメッセージのポスト先ウィンドウハンドルを指定します。
Visual Basicをご使用の場合は、0を指定して下さい。
メッセージポストを行わない場合には、NULLを指定して下さい。

BoardNo　・・・ ボード番号を指定します。0～9のいずれかになります。

PortNo　・・・ I/O PORT番号を指定します。

・C-870v1、C-874、C-874v1の場合

引数の*PortNo*値	意味
MPL_PORT	汎用I/O PORTをオープンします。

・C-872の場合

引数の*PortNo*値	意味
MPL_PORT1	汎用I/O PORT1をオープンします。
MPL_PORT2	汎用I/O PORT2をオープンします。

・C-875の場合

引数の*PortNo*値	意味
MPL_PORT	汎用I/O PORTをオープンします。
MPL_EX_PORT_IN10	増設I/O PORT IN10をオープンします。
MPL_EX_PORT_IN20	増設I/O PORT IN20をオープンします。
MPL_EX_PORT_IN30	増設I/O PORT IN30をオープンします。
MPL_EX_PORT_IN40	増設I/O PORT IN40をオープンします。
MPL_EX_PORT_OUT10	増設I/O PORT OUT10をオープンします。
MPL_EX_PORT_OUT20	増設I/O PORT OUT20をオープンします。
MPL_EX_PORT_OUT30	増設I/O PORT OUT30をオープンします。
MPL_EX_IO_INT_SET_PORT	増設I/O INT SET PORTをオープンします。
MPL_EX_IO10_20_INT_STATUS_PORT	増設I/O IN10_20 INT STATUS PORTをオープンします。
MPL_EX_IO30_40_INT_STATUS_PORT	増設I/O IN30_40 INT STATUS PORTをオープンします。
MPL_EX_PORT_IN10_IN20	増設I/O PORT IN10_1N20をオープンします。
MPL_EX_PORT_IN30_IN40	増設I/O PORT IN30_1N40をオープンします。

(注1)IOINT出力(IN10INT、IN20INT、IN30INT、IN40INT)による割り込みを使用する場合、
次に示すI/O PORT番号を指定して、I/O PORTをオープンして下さい。
IN10INT ・・・ MPL_EX_PORT_IN10_IN20　　IN30INT ・・・ MPL_EX_PORT_IN30_IN40
IN20INT ・・・ MPL_EX_PORT_IN10_IN20　　IN40INT ・・・ MPL_EX_PORT_IN30_IN40

(注2)MPL_EX_IO_INT_SET_PORT、MPL_EX_IO10_20_INT_STATUS_PORT、MPL_EX_IO30_40_INT_STATUS_
PORTをオープンした場合、割り込み機能が動作保障されませんので、御注意ください。

phPort　・・・ I/O PORTハンドルが格納される変数のポインタを指定します。

psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

I/O PORTクローズ関数

機　能

I/O PORTをクローズします。

書　式

C言語　BOOL MPL_BPortClose(DWORD hPort, MPL_S_RESULT FAR *psResult);

VB　Function MPL_BPortClose(ByVal hPort As Long, psResult As MPL_S_RESULT) As Boolean

Delphi　function MPL_BPortClose(hPort:DWORD; var psResult:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BPortClose(ByVal hPort As Integer, ByRef psResult As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BPortClose(uint hPort, ref MPL_S_RESULT psResult);

引　数

hPort ･･･ I/O PORTハンドルを指定します。
psResult ･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

I/O PORT読み出し関数

機　能

I/O PORTの内容を読み出します。

書　式

C言語　BOOL MPL_BPortIn(DWORD hPort, WORD FAR* pData, MPL_S_RESULT FAR* psResult);

VB　Function MPL_BPortIn(ByVal hPort As Long, pData As Integer, psResult As MPL_S_RESULT)
As Boolean

Delphi　function MPL_BPortIn(hPort:DWORD; var pData:WORD; var psResult:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BPortIn(ByVal hPort As Integer, ByRef pData As Short,
ByRef psResult As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BPortIn(uint hPort, ref ushort pData, ref MPL_S_RESULT psResult);

引　数

hPort ･･･ I/O PORTハンドルを指定します。
pData ･･･ 読み出した内容が格納される変数のポインタを指定します。
I/O PORTオープン拡張関数でMPL_EX_PORT_IN10_IN20又は、MPL_EX_PORT_IN30_IN40を指定して
I/O PORTをオープンした後に当I/O PORT読み出し関数を実行した場合、読み出した内容は16ビット
が有効です。(上位8ビット:IN20/IN40、下位8ビット:IN10/IN30)
他の場合は、読み出した内容は下位8ビットが有効です。上位8ビットは、0が読み出されます。

MPL_EX_PORT_IN10_IN20(MPL_EX_PORT_IN30_IN40)を指定した場合

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2^7	←		*IN20* (*IN40*)			→	2^0	2^7	←		*IN10* (*IN30*)			→	2^0

psResult ･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

I/O PORT書き込み関数

機　能

I/O PORTにデータを書き込みます。

書　式

C言語　BOOL MPL_BPortOut(DWORD *hPort*, WORD FAR* *pData*, MPL_S_RESULT FAR* *psResult*);

VB　Function MPL_BPortOut(ByVal *hPort* As Long, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BPortOut(*hPort*:DWORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_BPortOut(ByVal *hPort* As Integer, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BPortOut(uint *hPort*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

hPort ・・・ I/O PORTハンドルを指定します。
pData ・・・ 書き込むデータが格納されている変数のポインタを指定します。
書き込むデータが格納されている変数は下位8ビットが有効です。
psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

任意I/O PORT読み出し関数

機　能

ボード番号を指定してI/O PORTを読み出します。
C-872の場合、汎用I/O PORT1の内容を読み出します。

書　式

C言語　BOOL MPL_Inp(WORD *BoardNo*, WORD FAR* *pData*, MPL_S_RESULT FAR* *psResult*);

VB　Function MPL_Inp(ByVal *BoardNo* As Integer, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_Inp(*BoardNo*:WORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_Inp(ByVal *BoardNo* As Short, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.Inp(ushort *BoardNo*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

BoardNo ・・・ ボード番号(0～9)を指定します。
pData ・・・ 読み出した内容が格納される変数のポインタを指定します。
読み出した内容は下位8ビットが有効です。上位8ビットは、0が読み出されます。
psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

任意I/O PORT読み出し拡張関数

機　能

ボード番号、PORT番号を指定してI/O PORTを読み出します。

書　式

C言語　BOOL MPL_InpEx(WORD *BoardNo*, WORD *PortNo*, WORD FAR* *pData*, MPL_S_RESULT FAR* *psResult*);

VB　Function MPL_InpEx(ByVal *BoardNo* As Integer, ByVal *PortNo* As Integer, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_InpEx(*BoardNo*:WORD; *PortNo*:WORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT): Boolean;

VB.NET　Function MPL_InpEx(ByVal *BoardNo* As Short, ByVal *PortNo* As Short, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.InpEx(ushort *BoardNo*, ushort *PortNo*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

BoardNo ･･･ ボード番号(0～9)を指定します。
PortNo ･･･ I/O PORT番号を指定します。

・C-870v1、C-874、C-874v1の場合

引数の*PortNo*の値	意味
MPL_PORT	汎用I/O PORTを読み出します。

・C-872の場合

引数の*PortNo*の値	意味
MPL_PORT1	汎用I/O PORT1を読み出します。
MPL_PORT2	汎用I/O PORT2を読み出します。

・C-875の場合

引数の*PortNo*の値	意味
MPL_PORT	汎用I/O PORTを読み出します。
MPL_EX_PORT_IN10	増設I/O PORT IN10 PORT(IN10～IN17)を読み出します。
MPL_EX_PORT_IN20	増設I/O PORT IN20 PORT(IN20～IN27)を読み出します。
MPL_EX_PORT_IN30	増設I/O PORT IN30 PORT(IN30～IN37)を読み出します。
MPL_EX_PORT_IN40	増設I/O PORT IN40 PORT(IN40～IN47)を読み出します。
MPL_EX_PORT_OUT10	増設I/O PORT OUT10 PORT(OUT10～OUT17)を読み出します。
MPL_EX_PORT_OUT20	増設I/O PORT OUT20 PORT(OUT20～OUT27)を読み出します。
MPL_EX_PORT_OUT30	増設I/O PORT OUT30 PORT(OUT30～OUT37)を読み出します。
MPL_EX_IO10_20_INT_STATUS_PORT	増設IN10_IN20 INT STATUS PORTを読み出します。
MPL_EX_IO30_40_INT_STATUS_PORT	増設IN30_IN40 INT STATUS PORTを読み出します。
MPL_EX_PORT_IN10_IN20	増設I/O PORT IN10 PORT(IN10～IN17)、増設I/O PORT IN20 PORT(IN20～IN27)を読み出します。
MPL_EX_PORT_IN30_IN40	増設I/O PORT IN30 PORT(IN30～IN37)、増設I/O PORT IN40 PORT(IN40～IN47)を読み出します。

pData ･･･ 読み出した内容が格納される変数のポインタを指定します。
C-870v1、C-872、C-874、C-874v1の場合、読み出した内容は下位8ビットが有効です。
上位8ビットは、0が読み出されます。
C-875の場合、I/O PORT番号にMPL_EX_PORT_IN10_IN20又は、MPL_EX_PORT_IN30_IN40を指定時のみ、読み出した内容は16ビットが有効です。(上位8ビット:IN20/IN40、下位8ビット:IN10/IN30)
※.読み出した内容の詳細は、I/O PORT読み出し関数を御参照下さい。
他のI/O PORT番号を指定時は、読み出した内容は下位8ビットが有効です。
上位8ビットは、0が読み出されます。
psResult ･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

任意I/O PORT書き込み関数

機　能

ボード番号を指定してI/O PORTにデータを書き込みます。
C-872の場合、汎用I/O PORT1にデータを書き込みます。

書　式

C言語　BOOL MPL_Outp(WORD *BoardNo*, WORD FAR* *pData*, MPL_S_RESULT FAR* *psResult*);

VB　Function MPL_Outp(ByVal *BoardNo* As Integer, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_Outp(*BoardNo*:WORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_Outp(ByVal *BoardNo* As Short, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.Outp(ushort *BoardNo*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

BoardNo ・・・ ボード番号(0～9)を指定します。
pData ・・・ 書き込むデータが格納されている変数のポインタを指定します。
書き込むデータが格納されている変数は下位8ビットが有効です。
psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

任意I/O PORT書き込み拡張関数

機　能

ボード番号、PORT番号を指定してI/O PORTにデータを書き込みます。
※.IOINTを割り込みとして処理するプログラムでは、当関数のご使用はお避け下さい。

書　式

C言語　BOOL MPL_OutpEx(WORD *BoardNo*, WORD *PortNo*, WORD FAR* *pData*, MPL_S_RESULT FAR* *psResult*);

VB　Function MPL_OutpEx(ByVal *BoardNo* As Integer, ByVal *PortNo* As Integer, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_OutpEx(*BoardNo*:WORD; *PortNo*:WORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_OutpEx(ByVal *BoardNo* As Short, ByVal *PortNo* As Short, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.OutpEx(ushort *BoardNo*, ushort *PortNo*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

BoardNo　･･･ ボード番号(0～9)を指定します。
PortNo　･･･ I/O PORT番号を指定します。

・C-870v1、C-874、C-874v1の場合

引数の*PortNo*の値	意味
MPL_PORT	汎用I/O PORTにデータを書き込みます。

・C-872の場合

引数の*PortNo*の値	意味
MPL_PORT1	汎用I/O PORT1にデータを書き込みます。
MPL_PORT2	汎用I/O PORT2にデータを書き込みます。

・C-875の場合

引数の*PortNo*の値	意味
MPL_PORT	汎用I/O PORTにデータを書き込みます。
MPL_EX_PORT_OUT10	増設I/O PORT OUT10にデータを書き込みます。
MPL_EX_PORT_OUT20	増設I/O PORT OUT20にデータを書き込みます。
MPL_EX_PORT_OUT30	増設I/O PORT OUT30にデータを書き込みます。
MPL_EX_IO_INT_SET_PORT	増設I/O INT SET PORTにデータを書き込みます。

pData　･･･ 書き込むデータが格納されている変数のポインタを指定します。
書き込むデータが格納されている変数は下位8ビットが有効です。
psResult ･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

Z/A SENSOR割り付けPORT読み出し関数

機　能

ボード番号を指定してC-874v1のZ/A SENSOR割り付けPORTのデータを読み出します。

書　式

C言語　BOOL MPL_BRSensor(WORD *BoardNo*, WORD *PortNo*, WORD FAR* *pData*, MPL_S_RESULT FAR* *psResult*);

VB　Function MPL_BRSensor(ByVal *BoardNo* As Integer, ByVal *PortNo* As Integer, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BRSensor(*BoardNo*:WORD; *PortNo*:WORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT): Boolean;

VB.NET　Function MPL_BRSensor(ByVal *BoardNo* As Short, ByVal *PortNo* As Short, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BRSensor(ushort *BoardNo*, ushort *PortNo*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

BoardNo ・・・ ボード番号(0～9)を指定します。

PortNo ・・・ SENSOR割り付けPORTを指定します。

引数の*PortNo*の値	意味
MPL_Z_SENSOR_PORT	Z.SENSOR割り付けPORT
MPL_A_SENSOR_PORT	A.SENSOR割り付けPORT

pData ・・・ 読み出した内容が格納される変数のポインタを指定します。
読み出した内容は下位8ビットが有効です。上位8ビットは、0が読み出されます。

psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

Z/A SENSOR割り付けPORT書き込み関数

機　能

ボード番号を指定してC-874v1のZ/A SENSOR割り付けPORTにデータを書き込みます。

書　式

C言語　BOOL MPL_BWSensor(WORD *BoardNo*, WORD *PortNo*, WORD FAR* *pData*, MPL_S_RESULT FAR* *psResult*);

VB　Function MPL_BWSensor(ByVal *BoardNo* As Integer, ByVal *PortNo* As Integer, *pData* As Integer, *psResult* As MPL_S_RESULT) As Boolean

Delphi　function MPL_BWSensor(*BoardNo*:WORD; *PortNo*:WORD; var *pData*:WORD; var *psResult*:MPL_S_RESULT): Boolean;

VB.NET　Function MPL_BWSensor(ByVal *BoardNo* As Short, ByVal *PortNo* As Short, ByRef *pData* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.BWSensor(ushort *BoardNo*, ushort *PortNo*, ref ushort *pData*, ref MPL_S_RESULT *psResult*);

引　数

BoardNo ・・・ ボード番号(0～9)を指定します。

PortNo ・・・ SENSOR割り付けPORTを指定します。

引数の*PortNo*の値	意味
MPL_Z_SENSOR_PORT	Z.SENSOR割り付けPORT
MPL_A_SENSOR_PORT	A.SENSOR割り付けPORT

pData ・・・ 書き込むデータが格納されている変数のポインタを指定します。
書き込むデータが格納されている変数は下位8ビットが有効です。

psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

デバイス割り込みオープン関数

機　能

デバイス割り込み設定構造体をもとに指定デバイスの割り込みをオープンします。
※.当関数を実行する前に、割り込みINITIALIZE関数を実行して下さい。

書　式

C言語　BOOL MPL_InterruptOpen(DWORD *hDev*, MPL_S_INT_CONFIG FAR* *psIntConfig*, MPL_S_RESULT FAR* *psResult*);

Delphi　function MPL_InterruptOpen(*hDev*:DWORD; var *psIntConfig*:MPL_S_INT_CONFIG; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_InterruptOpen(ByVal *hDev* As Integer, ByRef *psIntConfig* As MPL_S_INT_CONFIG, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.InterruptOpen(uint *hDev*, ref MPL_S_INT_CONFIG *psIntConfig*, ref MPL_S_RESULT *psResult*);

引　数

hDev ・・・ デバイスハンドルを指定します。
psIntConfig ・・・ デバイス割り込み設定構造体のポインタを指定します。
psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

デバイス割り込みクローズ関数

機　能

指定されたデバイスの割り込みをクローズします。

書　式

C言語　BOOL MPL_InterruptClose(DWORD *hDev*, MPL_S_RESULT FAR* *psResult*);

Delphi　function MPL_InterruptClose(*hDev*:DWORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_InterruptClose(ByVal *hDev* As Integer, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.InterruptClose(uint *hDev*, ref MPL_S_RESULT *psResult*);

引　数

hDev ・・・ デバイスハンドルを指定します。
psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

I/O PORT割り込みオープン関数

機　能

I/O PORT INT SET構造体、I/O PORT割り込み設定構造体をもとに指定I/O PORTの割り込みをオープンします。
※.当関数を実行する前に、割り込みINITIALIZE関数を実行して下さい。

書　式

C言語　BOOL MPL_IoPortInterruptOpen(DWORD *hPort*, MPL_S_IO_PORT_INT_SET FAR **psIoPortIntSet*, MPL_S_IO_PORT_INT_CONFIG_INFO FAR **psIoPortIntConfig*, MPL_S_RESULT FAR **psResult*);

Delphi　function MPL_IoPortInterruptOpen(*hPort*:DWORD; var *psIoPortIntSet*:MPL_S_IO_PORT_INT_SET; var *psIoPortIntConfig*:MPL_S_IO_PORT_INT_CONFIG_INFO; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_IoPortInterruptOpen (ByVal *hPort* As Integer, ByRef *psIoPortIntSet* As MPL_S_IO_PORT_INT_SET, ByRef *psIoPortIntConfig* As MPL_S_IO_PORT_INT_CONFIG_INFO, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.IoPortInterruptOpen(uint *hPort*, ref MPL_S_IO_PORT_INT_SET *psIoPortIntSet*, ref MPL_S_IO_PORT_INT_CONFIG_INFO *psIoPortIntConfig*, ref MPL_S_RESULT *psResult*);

引　数

hPort ・・・ I/O PORTハンドルを指定します。
psIoPortIntSet ・・・ I/O PORT INT SET構造体のポインタを指定します。
psIoPortIntConfig ・・・ I/O PORT割り込み設定構造体のポインタを指定します。
psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

I/O PORT割り込みクローズ関数

機　能

指定されたI/O PORTの割り込みをクローズします。

書　式

C言語　BOOL MPL_IoPortInterruptClose(DWORD *hPort*, MPL_S_RESULT FAR* *psResult*);

Delphi　function MPL_IoPortInterruptClose(*hPort*:DWORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_IoPortInterruptClose(ByVal *hPort* As Integer, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.IoPortInterruptClose(uint *hPort*, ref MPL_S_RESULT *psResult*);

引　数

hPort ・・・ I/O PORTハンドルを指定します。
psResult ・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

割り込みイニシャライズ関数

機　能

指定されたBOARD CONTROLLERの割り込みを初期状態にします。詳細は割り込みを御参照下さい。

書　式

C言語　BOOL MPL_IntInitialize(WORD *BoardNo*, MPL_S_RESULT FAR **psResult*);

Delphi　function MPL_IntInitialize(*BoardNo*:WORD; var *psResult*:MPL_S_RESULT):Boolean;

VB.NET　Function MPL_IntInitialize(ByVal *BoardNo* As Short, ByRef *psResult* As MPL_S_RESULT) As Boolean

C#.NET　bool MPL.IntInitialize(ushort *BoardNo*, ref MPL_S_RESULT *psResult*);

引　数

BoardNo　・・・ ボード番号(0～9)を指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

データセット関数

機　能

32ビットデータを次の形式でデータ構造体に格納します。
*Data*の2^{31}～2^{24}は、無視します。

引数*psData*で示されるデータ構造体のメンバ*MPL_Data[0]*（C言語表記）［各種DATA1 PORTに対応］

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
0	0	0	0	0	0	0	0	2^{23}	←　引数*Data*の2^{23}～2^{16}　→						2^{16}

引数*psData*で示されるデータ構造体のメンバ*MPL_Data[1]*（C言語表記）［各種DATA2 PORTに対応］

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
0	0	0	0	0	0	0	0	2^{15}	←　引数*Data*の2^{15}～2^{8}　→						2^{8}

引数*psData*で示されるデータ構造体のメンバ*MPL_Data[2]*（C言語表記）［各種DATA3 PORTに対応］

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
0	0	0	0	0	0	0	0	2^{7}	←　引数*Data*の2^{7}～2^{0}　→						2^{0}

書　式

C言語　`VOID MPL_SetData(DWORD Data, MPL_S_DATA FAR *psData);`

VB　`Sub MPL_SetData(ByVal Data As Long, psData As MPL_S_DATA)`

Delphi　`procedure MPL_SetData(Data:DWORD; var psData:MPL_S_DATA);`

VB.NET　`Sub MPL_SetData(ByVal Data As Integer, ByRef psData As MPL_S_DATA)`

C#.NET　`void MPL.SetData(int Data, ref MPL_S_DATA psData);`

引　数

Data　　･･･ 32ビットのデータを指定します。
psData　･･･ データ構造体のポインタを指定します。

戻り値

この関数に、戻り値はありません。

データゲット関数

機　能

データ構造体の内容を次の形式で32ビットデータに変換し返します。

変換後の32ビットデータ

2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
下位24ビットを符号拡張								2^7	← メンバ*MPL_Data[0]* [各種DATA1 PORTに対応] →						2^0

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2^7	← メンバ*MPL_Data[1]* [各種DATA2 PORTに対応] →						2^0	2^7	← メンバ*MPL_Data[2]* [各種DATA3 PORTに対応] →						2^0

メンバはC言語表記です

書　式

C言語　DWORD MPL_GetData(MPL_S_DATA FAR **psData*);

VB　Function MPL_GetData(*psData* As MPL_S_DATA) As Long

Delphi　function MPL_GetData(var *psData*:MPL_S_DATA):DWORD;

VB.NET　Function MPL_GetData(ByRef *psData* As MPL_S_DATA) As Integer

C#.NET　int MPL.GetData(ref MPL_S_DATA *psData*);

引　数

psData　･･･ データ構造体のポインタを指定します。

戻り値

32ビットに変換されたデータを返します。

7. ソフト開発に必要なファイル

ユーザアプリケーション開発に必要なファイルは、インストール時に指定する次のフォルダに格納されています。
（インストール時にパスを¥Program Files指定した場合）

(1)Visual C++.NET、Visual C++
ヘッダファイル　　　　　¥Program Files¥Mpl16v5¥Bin¥Vc¥MplB.h
ライブラリファイル　　　¥Program Files¥Mpl16v5¥Bin¥Vc¥VcMplB.lib

(2)Visual Basic
関数定義ファイル　　　　¥Program Files¥Mpl16v5¥Bin¥Vb¥MplB.bas

(3)Delphi
関数定義ファイル　　　　¥Program Files¥Mpl16v5¥Bin¥Delphi¥MplB.pas

(4)C++ Builder
ヘッダファイル　　　　　¥Program Files¥Mpl16v5¥Bin¥Builder¥MplB.h
ライブラリファイル　　　¥Program Files¥Mpl16v5¥Bin¥Builder¥BcMplB.lib

(5)Visual Basic.NET
関数定義ファイル　　　　¥Program Files¥Mpl16v5¥Bin¥Vb.NET¥MplB.vb

(6)C#.NET
関数定義ファイル　　　　¥Program Files¥Mpl16v5¥Bin¥C#.NET¥MplB.cs

8. サンプルプログラムについて

Visual C++.NET、Visual C++、Delphi、C++ Builder、Visual Basic、Visual Basic.NET、C#.NETのサンプルプログラムが用意されています。
サンプルのファイルは、インストール時に指定する次のフォルダに格納されています。
(インストール時にパスを¥Program Files指定した場合)

(1)Visual C++.NET、Visual C++
　¥Program Files¥Mpl16v5¥Sample¥Vc
(2)Delphi
　¥Program Files¥Mpl16v5¥Sample¥Delphi
(3)C++ Builder
　¥Program Files¥Mpl16v5¥Sample¥Builder
(4)Visual Basic
　¥Program Files¥Mpl16v5¥Sample¥Vb
(5)Visual Basic.NET
　¥Program Files¥Mpl16v5¥Sample¥Vb.NET
(6)C#.NET
　¥Program Files¥Mpl16v5¥Sample¥C#.NET

8-1.仕様

使用するボードのボード番号を0に設定してください。
本サンプルプログラムはボード番号0のＸ軸でのみ動作します。
サンプルプログラムには、C#.NET 2002、Visual Basic.NET 2002、Visual C++6.0、C++ Builder5、Visual Basic6.0、Delphi5で作成したものを用意してあります。
これらは同じ仕様で作られています。
サンプルプログラムを参照する場合には、それぞれの言語の開発環境からプロジェクトファイルを開いてください。

Openボタン	・・・	デバイスをオープンします。
Closeボタン	・・・	デバイスをクローズします。
Endボタン	・・・	サンプルプログラムを終了します。
Resetボタン	・・・	ADDRESS COUNTERを0にPRESETします。
Stopボタン	・・・	DRIVEを即時停止します。
Data Setボタン	・・・	次の設定にします。 DRIVE TYPE ： L-TYPE　　LSPD ： 1000Hz URATE ： 10ms/1000Hz　　HSPD ： 5000Hz DRATE ： 10ms/1000Hz
Scan Drive +ボタン	・・・	+(CW)方向へSCAN DRIVEします。
Scan Drive -ボタン	・・・	-(CCW)方向へSCAN DRIVEします。
Org Drive ボタン	・・・	機械原点検出形式ORG-3でORIGIN DRIVEを行います。
Rtn Driveボタン	・・・	絶対ADDRESS 0へ移動するABSOLUTE INDEX DRIVEを行います。
Index Drive +ボタン	・・・	+(CW)方向へ3000パルス移動するINCREMENTAL INDEX DRIVEを行います。
Index Drive -ボタン	・・・	-(CCW)方向へ3000パルス移動するINCREMENTAL INDEX DRIVEを行います。
Demoボタン	・・・	次の動作を連続して行います。 ① 機械原点の検出(ORIGIN DRIVE) ② 電気原点の設定(ADDRESS COUNTERを0にPRESET) ③ +(CW)方向へ4000パルス移動を4回繰り返す(INCREMENTAL INDEX DRIVE) ④ 絶対ADDRESS 30000へ移動(ABSOLUTE INDEX DRIVE) ⑤ 電気原点(絶対ADDRESS 0)へ移動(ABSOLUTE INDEX DRIVE)

9.トラブルシューティング

9-1.作成したアプリケーションプログラムが正常に動作しない場合

①信号チェックプログラムを起動し、次のことを確認してください。
信号チェックプログラムは、起動時にボードのチェックを行っています。
問題がある場合、次のメッセージを表示します。

・メッセージ：Even 1 sheets of board isn't able to detect it.
　原因　　　：ボードがPCIバスに接続されていないことが考えられます。
・メッセージ：The detected board is over 10 sheets
　原因　　　：ボードが10枚以上接続されていることが考えられます。
・メッセージ：The board number is doubling.
　原因　　　：複数のボードで同じボード番号が設定されている事が考えられます。

②信号チェックプログラムを使用して、LIMIT信号、FSSTOP信号の状態を確認してください。

9-2.アプリケーションプログラムのデバッグ

MPLの各関数は、アプリケーションプログラムによって与えられた引数の内容をチェックし、エラーがある場合は、FALSE(0)を返し、正常である場合は、TRUE(1)を返します。MPL関数が正常に動作していないと思われるステップの後にブレークポイントを設定し、MPL関数が返した値がTRUE(1)であることを確認してください。
TRUE(1)でない場合(FALSE(0))は、エラー原因を特定する為にRESULT構造体の内容を参照してください。

9-3.割り込みが発生しない場合、次のことが考えられます。

・割り込みINITIALIZE関数が実行されていない。
・各デバイスの割り込み要求信号(RDYINT,CNTINT,DFLINT)が出力される設定になっていない。
・各I/O PORTの割り込み要求信号(IOINT)が出力される設定になっていない。
・CNTINT、DFLINTの場合、CNTINT/DFLINT OUTPUT TYPEがラッチ出力、CNTINT/DFLINT LATCH TRIGGER TYPEがエッジラッチになっていない。
・デバイス割り込み設定構造体にコールバック関数、又は、メッセージを設定していない。
・I/O PORT割り込み設定構造体にコールバック関数、又は、メッセージを設定していない。
・メッセージを使用するのに、デバイスオープン関数で、ウィンドウ ハンドルを引数に渡していない。
・メッセージを使用するのに、I/O PORTオープンEx関数で、ウィンドウ ハンドルを引数に渡していない。

■ 製品保証

保証期間と保証範囲について

- 納入品の保証期間は、納入後１ヶ年と致します。
- 上記保証期間中に当社の責により故障を生じた場合は、その修理を当社の責任において行います。（日本国内のみ）
ただし、次に該当する場合は、この保証対象範囲から除外させて頂きます。
(1) お客様の不適当な取り扱い、ならびに使用による場合。
(2) 故障の原因が、当製品以外からの事由による場合。
(3) お客さまの改造、修理による場合。
(4) 製品出荷当時の科学・技術水準では予見が不可能だった事由による場合。
(5) その他、天災、災害等、当社の責にない場合。
(注1)ここでいう保証は、納入品単体の保証を意味するもので、納入品の故障により誘発される損害はご容赦頂きます。
(注2)当社において修理済みの製品に関しましては、保証外とさせて頂きます。

技術相談のお問い合わせ

TEL.(042)664-5382 FAX.(042)666-5664
E-mail　s-support@melec-inc.com

販売に関するお問い合わせ

TEL.(042)664-5384 FAX.(042)666-2031

株式会社メレック 制御機器営業部
〒193-0834 東京都八王子市東浅川町516-10
URL:http://www.melec-inc.com

記載内容は、製品改良のため予告なく変更することがありますのでご了承ください。